(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102741814 B
(45)授权公告日 2015.11.25

(21)申请号 201180008327.8
(22)申请日 2011.02.04
(30)优先权数据
12/699,901 2010.02.04 US
(85)PCT国际申请进入国家阶段日
2012.08.03
(86)PCT国际申请的申请数据
PCT/US2011/023798 2011.02.04
(87)PCT国际申请的公布数据
WO2011/097518 EN 2011.08.11
(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州
(72)发明人 N·A·雅各布森 J·希恩 E·朱亚特
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 顾嘉运
(51)Int.Cl.
G06F 9/44(2006.01)
(56)对比文件
CN 101410803 A,2009.04.15,
US 2009/0133001 A1,2009.05.21,
US 2009/0249051 A1,2009.10.01,
CN 101211272 A,2008.07.02,
vmware.Vmware:“VMware ThinAPP Development Guide”.《http://www.vmware.com/files/pdf/VMware_ThinApp_Deployment_Guide.pdf》.2009,
审查员 钟容

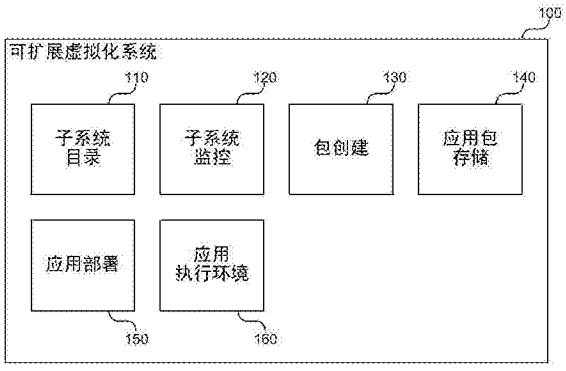
权利要求书2页 说明书8页 附图4页

(54)发明名称

可扩展应用虚拟化子系统

(57)摘要

一种可扩展虚拟化系统提供了一种对象模型，并以允许添加新的虚拟化子系统以便在不重制虚拟化产品本身的情况下扩展该虚拟化产品的方式管理虚拟应用生存期。虚拟化一应用一般涉及三个步骤：元数据提取、元数据存储和重构以及请求的运行时管理。该可扩展虚拟化系统接收应用准备会话的通知，并允许虚拟化子系统监控该会话以收集该应用用来在客户机上运行的、每一子系统专用的配置信息。每一子系统将收集到的信息提供给可扩展虚拟化系统，该可扩展虚拟化系统存储收集到的信息直到要虚拟化的应用被部署。当应用被部署时，该系统调用相同的虚拟化子系统并向该子系统提供所存储的信息。



CN 102741814 B

1. 一种用于监控应用以使得所述应用准备好进行虚拟化的计算机实现的方法，所述方法包括：

检测新进程的创建并将核心虚拟化系统注入所述进程中；

从虚拟化子系统的存储中加载已注册的虚拟化子系统，以允许所述核心虚拟化系统通过配置所述核心虚拟化系统并且在不修改和重建所述核心虚拟化系统的情况下与新子系统一起工作；

创建用于存储与同检测到的进程相关联的经虚拟化的应用相关的配置信息的应用包；

从所述已注册的虚拟化子系统中选择虚拟化子系统；

将检测到的进程创建通知给所选虚拟化子系统并且确定所述所选虚拟化子系统是否想要处理所述进程；以及

当确定所述所选虚拟化子系统能够处理所述检测到的进程时，

在所述检测到的进程运行时收集与该进程相关联的子系统专用配置信息；并且

将收集到的配置信息存储在所创建的包中，

其中，前述步骤由至少一个处理器来执行。

2. 如权利要求 1 所述的方法，其特征在于，检测新进程的创建包括打开监控，检测桌面应用的启动，启动核心虚拟化引擎，并且将所述核心虚拟化系统注入所述进程中，以使得所述核心虚拟化系统能够监控所述进程的 API 调用。

3. 如权利要求 1 所述的方法，其特征在于，加载已注册的虚拟化子系统包括加载所存储的关于在操作系统注册表中向所述核心虚拟化系统注册的虚拟化子系统的信息。

4. 如权利要求 1 所述的方法，其特征在于，创建所述应用包包括创建用于存储子系统专用配置设置的可扩展标记语言 (XML) 清单。

5. 如权利要求 1 所述的方法，其特征在于，选择所述虚拟化子系统包括迭代通过每一个已注册的虚拟化子系统直到一个子系统指示它能够处理所述检测到的进程。

6. 如权利要求 1 所述的方法，其特征在于，所述所选虚拟化子系统包括调用创建函数来加载所述虚拟化子系统并且传递关于所述进程的子系统信息。

7. 如权利要求 1 所述的方法，其特征在于，收集子系统专用配置信息包括使用 web 服务器虚拟化子系统来收集 web 服务器配置信息。

8. 如权利要求 1 所述的方法，其特征在于，存储收集到的配置信息包括从所述虚拟化子系统接收对对象模型接口的调用。

9. 如权利要求 1 所述的方法，其特征在于，还包括将子系统标识符存储在所创建的包中，所述子系统标识符指示要用来执行所述经虚拟化的应用的虚拟化子系统。

10. 一种用于应用层虚拟化的计算机系统，所述系统包括：

处理器和存储器，所述处理器和存储器被配置成执行软件指令；

子系统目录组件，所述子系统目录组件被配置成从至少一个虚拟化子系统接收注册请求，其中注册允许核心虚拟化系统在不更新该核心虚拟化系统的指令的情况下使用所述虚拟化子系统来虚拟化应用；

子系统监控组件，所述子系统监控组件被配置成调用已注册的虚拟化子系统来监控正准备进行虚拟化的应用；

包创建组件，所述包创建组件被配置成创建用于存储与正准备进行虚拟化的应用相关的配置数据的包；

应用部署组件，所述应用部署组件被配置成将应用包部署到客户机计算机系统，使得所述客户机计算机系统能够调用经虚拟化的应用；以及

应用执行环境，所述应用执行环境被配置成提供所述经虚拟化的应用和所述客户机计算机系统的主机操作系统之间的间接层。

11. 如权利要求 10 所述的系统，其特征在于，所述子系统监控组件还被配置成调用指示对应用进程感兴趣的一个或多个虚拟化子系统，以使得所述子系统能够在所述应用运行时监控所述应用做出的特定于子系统的配置改变。

12. 如权利要求 10 所述的系统，其特征在于，所述包创建组件还被配置成在新应用正准备进行虚拟化时创建所述包，并提供供子系统将子系统专用配置信息存储在所述包中的 API。

13. 如权利要求 10 所述的系统，其特征在于，所述应用部署组件还被配置成通过调用与所述应用包相关联的虚拟化子系统来在客户机系统上执行应用专用配置。

14. 如权利要求 10 所述的系统，其特征在于，所述应用执行环境还被配置成调用已注册的虚拟化子系统来提供在运行时对所述经虚拟化的应用的特定于子系统的处理。

可扩展应用虚拟化子系统

[0001] 背景

[0002] 虚拟化指的是物理硬件对虚拟机的执行然后在虚拟机上运行操作系统和应用。虚拟机可以表示硬件功能的最小公分母或者可以表示易于为其准备操作系统和应用的公知配置。许多数据中心使用虚拟化来随着资源需求增长、为了维护周期、以及平衡物理服务器负载而能够容易地将虚拟机移动到新的物理硬件。虚拟化对于许多情形是有用的，但也可能施加由于许多虚拟机争用同一资源(例如，中央处理单元(CPU)、存储器和网络接口卡(NIC))而出现的限制。

[0003] 应用虚拟化在单个应用的层面提供虚拟环境，将该应用与底层 OS 隔离，类似于虚拟机将 OS 与底层硬件隔离的方式。例如，操作系统可本机地运行一些应用，而同时提供运行其它应用的虚拟环境。这可以允许操作系统例如运行为不同操作系统所设计的应用。应用虚拟化在主机操作系统中本机运行的应用和在虚拟环境中运行的应用之间模糊了对用户的区别。例如，两类应用可以并排出现在操作系统外壳所提供的任务栏或菜单中。例如，微软应用虚拟化(Application Virtualization (App-V))将应用转换成未安装且与其他应用不冲突的集中管理的虚拟服务。在物理环境中，每一应用依赖于其操作系统(OS)来获得某一范围的服务，包括存储器分配、设备驱动程序以及更多。应用及其 OS 之间的不兼容性可通过服务器虚拟化或是呈现虚拟化来解决——但 OS 的同一实例上所安装的两个应用之间的不兼容性是由应用虚拟化来解决的。

[0004] 应用虚拟化产品的开发者经常扩展产品中的应用虚拟化子系统。例如，开发者可能想要虚拟化操作系统中先前未被虚拟化的部分以增加应用隔离。使用 MICROSOFTWINDOWS 系统的一个示例是添加对虚拟化 COM+ 的支持。COM+ 是微软组件对象模型(COM)和微软事务服务器(MTS)的进化。在现有的应用虚拟化解决方案中，虚拟化子系统与产品紧密地耦合。在现有产品中，该耦合意味着添加对如同 COM+ 的新扩展点的支持涉及重大的产品重制。另外，子系统实现涉及大量专家经验以及对操作系统内部组件的理解(不仅仅是子系统专用知识)。

[0005] 概述

[0006] 本文描述了一种可扩展虚拟化系统，该系统提供了一种对象模型，并以允许添加新的虚拟化子系统以便在不重制虚拟化产品本身的情况下扩展该虚拟化产品的方式来管理虚拟应用生存期。虚拟化一应用一般涉及三个步骤：元数据提取、元数据存储和重构以及请求的运行时管理。该可扩展虚拟化系统接收应用准备会话的通知，并允许虚拟化子系统监控该会话以收集该应用用来在客户机上运行的、每一子系统专用的配置信息。每一子系统将收集到的信息提供给可扩展虚拟化系统，该可扩展虚拟化系统存储收集到的信息直到要虚拟化的应用被部署。当应用被部署时，该系统调用相同的虚拟化子系统并向该子系统提供所存储的信息。由此，该可扩展虚拟化系统提供了对许多类型的虚拟化子系统都有用的通用模型，该模型允许在较少地影响实现该系统的虚拟化产品的情况下更容易地扩展该产品。

[0007] 提供本概述以便以简化形式介绍将在以下详细描述中进一步描述的一些概念。本

概述并不旨在标识所要求保护主题的关键特征或必要特征，也不旨在用于限制所要求保护主题的范围。

[0008] 附图简述

[0009] 图 1 是示出在一个实施例中的可扩展虚拟化系统的各组件的框图。

[0010] 图 2 是示出一个实施例中的可扩展虚拟化系统监控应用的处理的流程图。

[0011] 图 3 是示出一个实施例中的可扩展虚拟化系统部署经虚拟化的应用的处理的流程图。

[0012] 图 4 是示出在一个实施例中的操作系统以及可扩展虚拟化系统的实现组件的框图。

[0013] 详细描述

[0014] 本文描述了一种可扩展虚拟化系统，该系统提供了一种对象模型，并以允许添加新的虚拟化子系统以便在不重制虚拟化产品本身的情况下扩展该虚拟化产品的方式管理虚拟应用生存期。虚拟化一应用一般涉及三个步骤：元数据提取、元数据存储和重构以及请求的运行时管理。元数据提取是知晓应用何时准备进行虚拟化以及监控该应用以提取配置信息的过程。配置信息可包括应用将其自身束缚于操作系统或其他应用的任何方式。该可扩展虚拟化系统接收应用准备会话的通知，并允许虚拟化子系统监控该会话以收集该应用用来在客户机上运行的、每一子系统专用的配置信息。每一子系统将收集到的信息提供给可扩展虚拟化系统，该可扩展虚拟化系统存储收集到的信息直到要虚拟化的应用被部署。当应用被部署时，该系统调用相同的虚拟化子系统并向该子系统提供所存储的信息。该子系统使用该信息以及来自客户机的信息来重构元数据，并根据该知识知晓如何执行用于虚拟化应用的子系统专用步骤。

[0015] 例如，如果管理员想要虚拟化微软因特网信息服务器（IIS）web 应用，则 IIS 使用‘应用池’标识符来管理 web 应用实例。在该实例中，可扩展虚拟化系统的监控部分将找到已注册的虚拟化子系统。在这种情况下，该系统将会将 IIS 虚拟化子系统注入新进程中。当创建一进程时，可扩展虚拟化系统接收一通知，并通过子系统专用方法检测到 IIS 正在注册 web 应用。另选地或另外地，该子系统可在安装该应用之前和之后获取配置信息的快照，并将差异存储为应用元数据。IIS 虚拟化子系统将提取诸如 web 应用池的名称等虚拟化 web 应用所需的信息，按需修改请求以便在另一客户机上运行（例如，通过确保应用池名称未被使用），并令该请求继续。当 web 应用被部署时，可扩展虚拟化系统将发现该虚拟应用依赖于 IIS 虚拟化子系统，加载该子系统，然后通知该子系统正在创建 web 应用。可扩展虚拟化系统将向 IIS 虚拟化子系统提供该子系统在监控期间收集到的关于 web 应用的信息。在运行时，当 IIS 创建具有该特定名称的 web 应用池时，IIS 虚拟化子系统将知道得足够多以找到并创建该特定虚拟应用。可使用相同的步骤来为各种其他类型的应用和子系统提供虚拟化。由此，该可扩展虚拟化系统提供了对许多类型的虚拟化子系统都有用的通用模型，该模型允许在较少地影响实现该系统的虚拟化产品的情况下更容易地扩展该产品。另外，使用该系统，虚拟化子系统创作者能够更多地聚焦于子系统专用行为，并且子系统实现涉及对由可扩展虚拟化系统处理的操作系统专用知识的较少了解。

[0016] 应用虚拟化子系统的主要责任是监控、虚拟化、注册、以及运行时虚拟化。这些中的每一个在本文中被分开地描述。

[0017] 监控涉及监控安装进程以检测对计算机系统做出的改变。在某些实施例中，被称为定序器的应用负责监控应用虚拟化阶段。某些子系统可挂钩应用编程接口(API)来监控该安装，而其他子系统可以仅仅比较安装前和安装后的计算机系统状态。前者的示例是虚拟服务子系统挂钩 CreateService()API 来检测新操作系统服务何时被安装程序添加。另一方面，在先前的示例中所描述的 IIS 子系统可通过比较安装之前和之后的 IIS 配置来收集其使用的信息。

[0018] 虚拟化涉及向准备应用来虚拟化的管理员示出应用做出的变化的可视指示。定序器以一系列标签显示监控期间检测到的改变。可扩展虚拟化系统将实现虚拟化的责任赋予每一子系统。每一子系统提供适于该子系统的用于虚拟化的用户界面。这消除了对在虚拟化阶段期间定序器具有每个子系统的显式知识的需要。子系统的虚拟化界面负责在可扩展虚拟化系统的虚拟化应用为它们创建的窗口中显示它们的结果。

[0019] 注册涉及准备客户机计算机系统来运行虚拟应用。尽管虚拟应用未被安装在客户机上，但一些信息被发布给该客户机以提供无缝用户体验(例如，使得应用在开始菜单中出现)。可扩展虚拟化系统将监控期间由子系统收集的信息发布到客户机上的合适的配置位置。例如，经虚拟化的 IISweb 应用可修改客户机的 IIS 配置以提供该应用的存在的客户机知识(使得用户能够启动该应用)。

[0020] 子系统与注册紧密相关的另一责任是配置。注册是执行一次的；然而配置可在子系统注册其组件之后发生许多次。注册的示例是创建 IIS 网站、应用和本文所述的应用池。配置项的示例是数据库连接串。如果应用的后端数据库被移动，则该值可能需要在应用被注册之后改变。子系统负责应用任何子系统专用的配置值，诸如要求应用子系统的专门知识的配置(例如可能需要通过调用专门 API 来设置)。

[0021] 运行时虚拟化指的是在经虚拟化的应用的运行时对函数的挂钩和改变函数的行为，使得应用能够访问其资源，就好像它被本地地安装在客户机上。运行时虚拟化也指做出关于特定系统进程(即不是虚拟应用的包的一部分的进程)是否应被虚拟化的决策。例如，IIS 子系统基于命令行上传递的应用池名称来确定 IIS 工作者进程是否应被虚拟化。

[0022] 图 1 是示出在一个实施例中的可扩展虚拟化系统的各组件的框图。系统 100 包括子系统目录组件 110、子系统监控组件 120、包创建组件 130、应用包存储 140、应用部署组件 150 以及应用执行环境 160。这些组件中的每一个都在此处进一步详细讨论。

[0023] 子系统目录组件 110 从至少一个虚拟化子系统接收注册请求，其中注册允许核心虚拟化系统在不更新该核心虚拟化系统的指令的情况下使用虚拟化子系统来虚拟化应用。例如，被安装到客户机计算机的 COM+ 虚拟化子系统可以向核心虚拟化系统注册其自身。当用户请求虚拟化 COM+ 应用时，核心虚拟化系统调用 COM+ 虚拟化子系统，COM+ 虚拟化子系统提供对 COM+ 应用的子系统专用监控以检测配置改变。另外，当经虚拟化的应用在客户机上运行时，核心虚拟化系统再次调用虚拟化子系统来执行对该应用的子系统专用运行时处理。

[0024] 子系统监控组件 120 调用已注册的虚拟化子系统来监控正准备进行虚拟化的应用。子系统监控组件 120 可迭代通过已注册的子系统，询问每一个子系统该子系统是否对处理当前应用感兴趣。组件 120 调用指示出对该应用进程感兴趣的一个或多个子系统，使得这些子系统能够监控该应用做出的特定于子系统的配置改变。子系统监控组件 120 可以

将应用的生存期通知给子系统，以使得各个子系统能够执行与该子系统相关的启动和关闭任务。例如，一些子系统可通过获取应用运行之前和之后的配置数据的快照来进行监控。

[0025] 包创建组件 130 创建用于存储与正准备进行虚拟化的应用有关的配置数据的包。包可包括各种容器文件格式，容器文件格式可包括压缩、验证、加密或其他处理，以便使得包更小或提供包的创作者的安全性或必然性。例如，包可以是 ZIP、CAB 或适用于在单个文件内存储许多文件和设置的其他档案文件格式。包创建组件 130 在新应用正准备进行虚拟化时创建包，并提供供子系统将子系统专用配置信息存储在该包中的 API。例如，当可扩展虚拟化系统调用子系统时，组件 130 可传递指向用于存储数据的接口的指针。

[0026] 应用包存储 140 在对正准备进行虚拟化的应用的监控与经虚拟化的应用到一个或多个客户机计算机系统的部署之间存储应用包。应用包存储 140 可包括各种存储介质，如文件系统、基于网络的存储、基于云的存储服务、数据库等等。

[0027] 应用部署组件 150 将应用包部署到客户机计算机系统，使得客户机计算机系统能够调用经虚拟化的应用。组件 150 可在客户机系统上执行应用专用配置，如添加文件类型关联、添加到应用包的链接、执行操作系统服务配置、以及添加 web 服务器配置信息。应用部署组件 150 可调用与应用包相关联的子系统来执行子系统专用注册任务，以使得经虚拟化的应用准备好在客户机计算机系统上运行。

[0028] 应用执行环境 160 提供了经虚拟化的应用和客户机计算机系统的主操作系统之间的间接层。包装器可以非常瘦，从而允许应用几乎本机地运行，诸如在应用被设计成在主操作系统上运行时。另选地或另外地，包装器可提供 API，并且满足为其它操作系统或操作系统版本设计的应用所预期的其它约束条件。因此，应用执行环境 160 向虚拟应用提供使用主机操作系统的可用资源来为其设计应用的环境。应用执行环境 160 还调用合适的子系统（或多个子系统）来提供在运行时对经虚拟化的应用的特定于子系统的处理。

[0029] 其上实现可扩展虚拟化系统的计算设备可包括中央处理单元、存储器、输入设备（例如，键盘和定点设备）、输出设备（例如，显示设备）和存储设备（例如，盘驱动器或其他非易失性存储介质）。存储器和存储设备是可以用实现或启用该系统的计算机可执行指令（如软件）来编码的计算机可读存储介质。此外，数据结构和消息结构可被存储或经由诸如通信链路上的信号等数据传输介质传送。可以使用各种通信链路，诸如因特网、局域网、广域网、点对点拨号连接、蜂窝电话网络等。

[0030] 该系统的实施例可以在各种操作环境中实现，这些操作环境包括个人计算机、服务器计算机、手持式或膝上型设备、多处理器系统、基于微处理器的系统、可编程消费电子产品、数码相机、网络 PC、小型计算机、大型计算机、包括任何上述系统或设备的分布式计算环境等。计算机系统可以是蜂窝电话、个人数字助理、智能电话、个人计算机、可编程消费电子设备、数码相机等。

[0031] 该系统可以在由一个或多个计算机或其他设备执行的诸如程序模块等计算机可执行指令的通用上下文中描述。一般而言，程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、数据结构等等。通常，程序模块的功能可在各个实施例中按需进行组合或分布。

[0032] 图 2 是示出一个实施例中的可扩展虚拟化系统监控应用的处理的流程图。开始于框 210，该系统检测新进程的创建并将核心虚拟化系统注入该进程中。例如，如果管理员打

开监控并启动桌面应用，则该系统可检测该桌面应用的启动，启动核心虚拟化引擎，并将核心虚拟化系统注入该进程中，使得例如该系统能够监控 API 调用以及该进程的其他行为。继续至框 220，系统从虚拟化子系统的列表中加载已注册的虚拟化子系统。例如，该系统可存储关于在操作系统注册表中向该系统注册的虚拟化子系统的信息。以此方式，该系统能够简单地通过配置系统并且在不修改和重建系统的情况下（例如，在不是每一次需要新子系统时都装运新的核发布的情况下）与新系统一起工作。

[0033] 继续至框 230，系统创建用于存储配置信息的应用包，该配置信息与同检测到的进程相关联的经虚拟化的应用相关。例如，包可包括可扩展标记语言（XML）文件或用于配置设置的其他容器。继续至框 240，系统选择第一个注册的虚拟化子系统。在后续迭代中，系统选择下一个注册的子系统。继续至框 250，系统将检测到的进程创建通知给所选虚拟化子系统，并确定该虚拟化子系统是否想要虚拟化该进程。例如，系统可调用创建函数来加载虚拟化子系统并且然后传递关于该进程的子系统信息。子系统可提供来自该函数的指示该子系统是否对虚拟化该进程感兴趣的返回值。不同类型的进程可由不同的子系统来虚拟化，并由此通常是单个子系统将指示对检测到的进程负责。

[0034] 继续至判定框 260，如果系统确定所选子系统能够处理检测到的进程，则该系统继续至框 270，否则该系统循环至框 240 以选择下一个虚拟化子系统。如果没有子系统能够处理该进程（未示出），则该系统完成并且该进程本机地运行（即，不被虚拟化）。继续至判定框 270，虚拟化子系统在检测到的进程运行时收集关于该进程的子系统专用配置信息。例如，子系统可检测该进程修改的文件和注册表键，以及其他配置改变，例如微软活动目录（MICROSOFT ACTIVE DIRECTORY）或 IIS 元数据库改变。继续至框 280，系统将收集到的配置信息存储在所创建的包中。系统可以向子系统提供接口以使得子系统能够对和从所创建的包串行化和并行化子系统专用信息。由此，包将对核心虚拟化系统共同的信息以及子系统专用信息存储在一个地方。

[0035] 图 3 是示出一个实施例中的可扩展虚拟化系统部署经虚拟化的应用的处理的流程图。开始于框 310，系统在客户机计算机系统上注册经虚拟化的应用的实例。例如，可扩展虚拟化系统可创建链接、文件类型关联或允许客户机的用户启动经虚拟化的应用的其他客户机入口。继续至框 320，系统接收指示出启动经虚拟化的应用的请求的应用执行指示。在桌面应用的情况下，该指示可来自使用操作系统外壳的用户。在服务器应用的情况下，该指示可来自通过网络（例如，用于访问由服务器主存的网页）或其他源接收的请求。

[0036] 继续至框 330，系统加载与接收到的应用执行指示相关联的应用包。例如，该指示可包括调用的链接，该链接指定应用 GUID 或其他标识符以使得系统能够找到和启动正确的经虚拟化的应用。系统还可从加载的包中检索核心虚拟化配置设置。继续至框 340，系统标识与应用包相关联的虚拟化子系统。例如，系统可以通过查询已注册的子系统来从进程命令行中确定该进程是否已被虚拟化。客户机可以通过进程在包中的路径、通过专门指包名称的命令行自变量、或通过查询每一个子系统来动态地确定进程是否应该被虚拟化。如果应被虚拟化，则所有子系统在虚拟进程中都可以是活动的。例如，系统可遵循类似于图 2 的步骤 240 到 260 的步骤以迭代通过已注册的子系统，并允许子系统基于可用信息来指示对虚拟化进程的偏好。

[0037] 继续至框 350，系统启动所标识的虚拟化子系统。例如，系统可定位与子系统相关

联的二进制可执行模块，加载该模块，并执行导致子系统启动的入口点函数。继续至框 360，系统将执行应用包的指示通知给所标识的虚拟化子系统。例如，系统可以在启动子系统时将指向包的数据的指针传递给该子系统。子系统可设置允许该子系统处理经虚拟化的应用的执行的子系统专用挂钩。

[0038] 继续至框 370，系统在所标识的子系统请求时从应用包中检索子系统专用信息。可扩展虚拟化系统可提供用于对和从应用包串行化和并行化信息的通用接口，并且子系统可使用该接口来在本文描述的监控阶段期间检索子系统存储在包中的设置。继续至框 380，系统使用所标识的子系统和检索到的子系统专用信息来运行经虚拟化的应用。例如，系统可允许在 CreateProcess 调用期间暂停以使得虚拟化环境能够加载的进程继续执行。在应用执行时，被所标识的虚拟化子系统放在适当位置的任何挂钩或其他间接过程允许子系统处理要求重定向或其他处理的任何应用请求，以准许虚拟化提供的隔离。在框 380 之后，这些步骤结束。

[0039] 图 4 是示出在一个实施例中的操作系统以及可扩展虚拟化系统的实现组件的框图。在某些实施例中，可扩展虚拟化系统使用操作系统驱动程序 430 以及若干用户模式组件来虚拟化应用。虚拟化驱动程序 430（如 sftplay.sys）在操作系统的内核层管理虚拟环境。该驱动程序的进程管理器组件维护针对虚拟环境映射的进程 ID。在进程被创建时，进程管理器组件自动地将子进程添加到父进程的虚拟环境。进程管理器组件还向用户模式虚拟化库 420 通知虚拟进程创建、终止以及何时没有进程在虚拟环境中运行。

[0040] 驱动程序 430 的一个角色是执行注册表和文件系统虚拟化。当使用虚拟化库 420 首次创建虚拟环境时，注册表和文件映射信息被上载到驱动程序 430。驱动程序 430 使用该信息来更改注册表和文件系统 API 的行为，以便对于虚拟应用 440 而言看上去就好像该应用被本地安装在客户机计算机系统上。

[0041] 用户模式虚拟化库 420（如 osguard.lib）包含用于管理虚拟化环境和进程的 API，且是用户模式和虚拟化驱动程序 430 之间的接口。虚拟化库 420 还具有内建于其中的虚拟子系统，包括虚拟文件系统和虚拟注册表，这些虚拟子系统跨各种可扩展子系统被使用。虚拟化库 420 被用于打包虚拟应用，以及在运行时也被使用。定序器 410 是管理员用来通过监控安装进程来打包虚拟应用的应用，而监听器（未示出）是在运行时管理虚拟应用的操作系统服务。

[0042] 虚拟化运行时模块（如 sftldr.dll）是在创建子进程时由虚拟化库 420 或其自己注入到每个虚拟进程中的库。Detours 库被用于挂钩虚拟应用 440 进程内的函数。某些调用被重定向到虚拟化驱动程序 430 以便进行注册表和文件系统虚拟化；而其他调用做出对虚拟化库 420 的 RPC 调用以执行它们各自的子系统虚拟化。

[0043] 在其初始化期间，虚拟化库 420 动态地加载注册表中列出的每个子系统模块，并存储指向它们的工厂接口（factory interface）的指针。当定序器 410 或监听器使用虚拟环境管理器来创建虚拟环境时，使用工厂来创建每个子系统的实例。子系统实例被存储在虚拟环境对象内，并可由定序器或监听器检索。

[0044] 许多子系统需要访问虚拟文件系统和注册表来执行它们的虚拟化。例如，虚拟 COM 和虚拟服务子系统都需要访问虚拟注册表键来分别创建虚拟 COM 对象以及启动虚拟服务。因此，在某些实施例中，虚拟注册表和文件系统子系统不是一般化的，而是核心组件。子系

统可通过经它们的监控和运行时接口传递给它们的虚拟环境对象来获得对这些核心子系统的访问。

[0045] 监控接口在清单 450 文件中存储它检测到的子系统改变。该清单 450 被传递到包括可视化接口和注册接口的其他接口，可视化接口显示该信息并允许该信息被编辑，而注册接口使用该信息来注册组件。

[0046] 在一些实施例中，可扩展虚拟化系统存储的清单是 XML 文件。XML 文件用声明性格式分层存储信息，并允许用于虚拟化的各种组件和子系统单独地存储该子系统需要的信息。在运行时，每一个组件或子系统都能够容易地从 XML 文件中定位并提取其自己存储的信息。

[0047] 在一些实施例中，可扩展虚拟化系统提供用于与虚拟化子系统交互的对象模型。该系统为每一个子系统责任提供接口，并且表示子系统的接口包装其他接口。监控和运行时各自使用两个接口，在经虚拟化的进程中使用的一个接口与由本文描述的虚拟化库使用的一个接口通信。以下是所得接口。

[0048]

```
class subsystem
{
public:
    // 返回子系统名称。
    virtual const std::wstring& name() const = 0;

    // 返回子系统的虚拟化接口。
    virtual subsystem_visualizer* visualizer() = 0;

    // 返回子系统的注册接口。
    virtual subsystem_registrator* registrator() = 0;

    // 返回子系统的监控接口。
    virtual subsystem_monitor* monitor() = 0;

    // 返回子系统的在虚拟进程中使用的监控接口。
    virtual subsystem_process_monitor* process_monitor() = 0;

    // 返回子系统的运行时接口。
    virtual subsystem_runtime* runtime() = 0;
```

[0049]

```
    // 返回子系统的在虚拟进程中使用的运行时接口。
    virtual subsystem_process_runtime* process_runtime() = 0;

    // 返回子系统的 VE 包装器接口。
    virtual subsystem_ve_wrapper* ve_wrapper() = 0;
};
```

[0050] 在一些实施例中，可扩展虚拟化系统从虚拟化子系统接收用于与各子系统一起工作的工厂接口。一些子系统需要将各进程共享的状态存储在虚拟环境中，从而使用工厂接口来为每一个虚拟环境创建子系统对象的实例。每一个子系统模块导出返回工厂对象的列

表的函数，每一个该模块实现的子系统一个工厂对象。在每一次创建虚拟环境时，虚拟化库使用各工厂来创建子系统的实例。配置信息（例如，注册表键）控制虚拟化库加载哪些子系统模块。以下是工厂接口定义。

[0051]

```
class subsystem_factory
{
public:
  // 返回该工厂将创建的子系统的名称。
  virtual const std::wstring& name() const = 0;

  // 创建子系统对象的实例。
  virtual shared_ptr<subsystem> create() const = 0;
};
```

[0052] 在一些实施例中，可扩展虚拟化系统调用虚拟化子系统以供测试。除了实现虚拟子系统之外，这些模块还可用于测试和调试。打包虚拟应用的最困难方面之一是在应用在虚拟环境中不恰当地运行时诊断问题。因为子系统模块被注入虚拟进程中，所以模块能够通过挂钩 API 和查找特定差错来监控虚拟进程。这些模块的另一用途是对虚拟化库和定序器进行功能测试。模块能够通过检查原始数据来在监控期间验证虚拟注册表并且文件系统捕捉到正确的信息。这些模块还可通过从它们的接口中生成差错来测试定序器和监听器的容错。

[0053] 从前面的描述中可以看出，可以理解，此处描述的可扩展虚拟化系统的特定实施例只是为了说明，但是，在不偏离本发明的精神和范围的情况下，可以进行各种修改。因此，本发明只受所附权利要求限制。

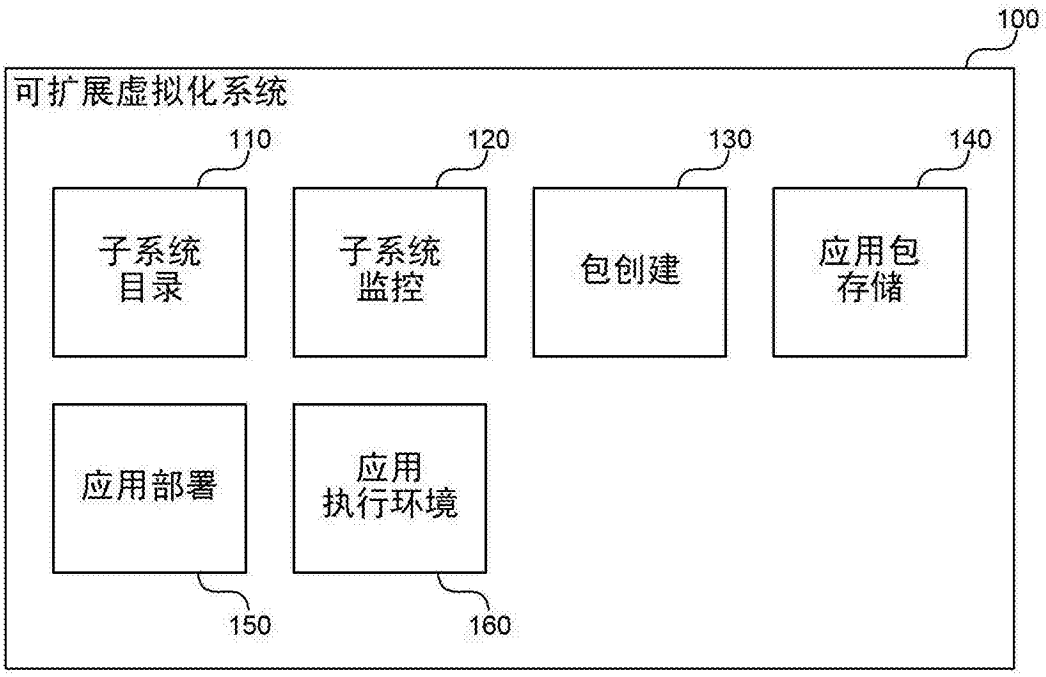

100
可扩展虚拟化系统
110
子系统
目录
120
子系统
监控
130
包创建
140
应用包
存储
应用部署
应用
执行环境
150
160

图 1

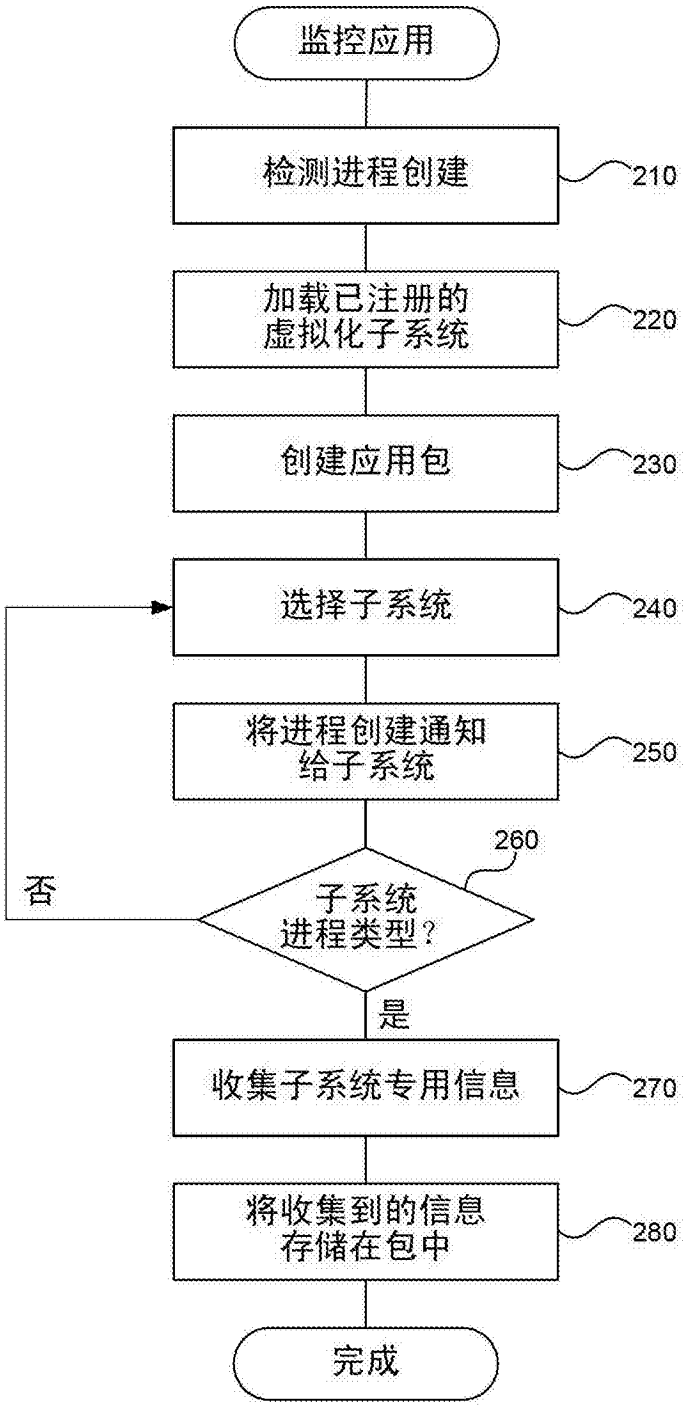

监控应用
检测进程创建
210
加载已注册的
虚拟化子系统
220
创建应用包
230
选择子系统
240
将进程创建通知
给子系统
250
260
子系统
进程类型?
否
是
收集子系统专用信息
270
将收集到的信息
存储在包中
280
完成

图 2

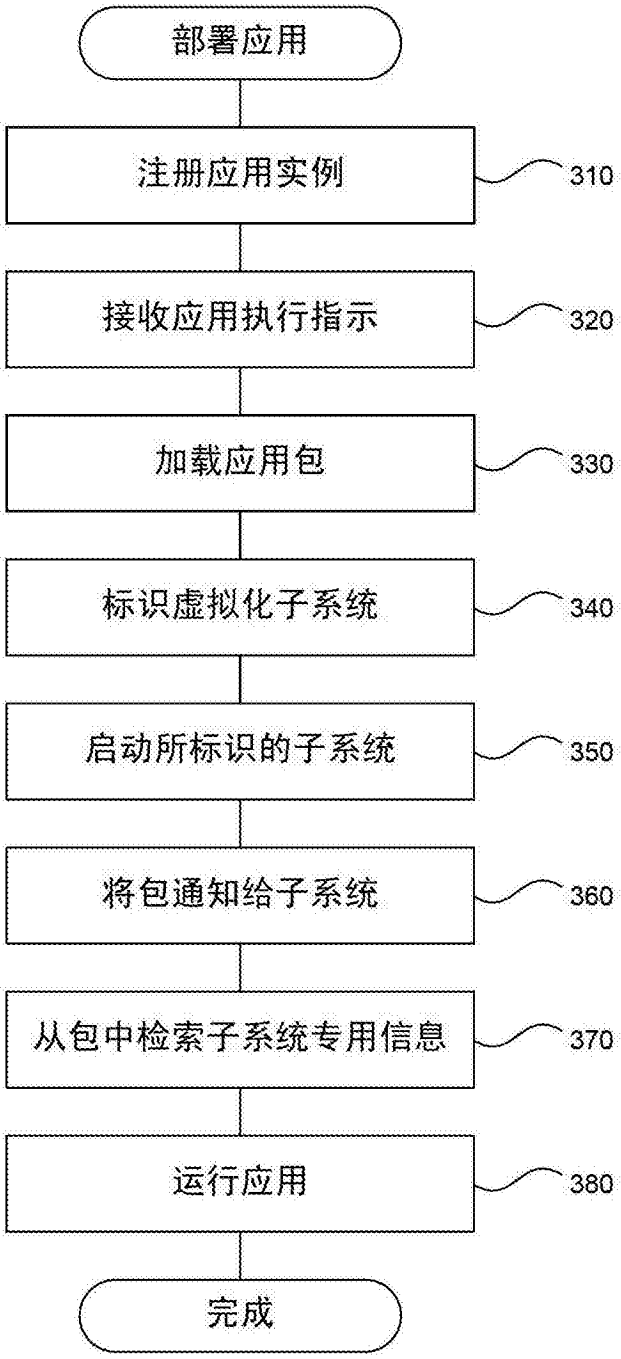

部署应用
注册应用实例
310
接收应用执行指示
320
加载应用包
330
标识虚拟化子系统
340
启动所标识的子系统
350
将包通知给子系统
360
从包中检索子系统专用信息
370
运行应用
380
完成

图 3

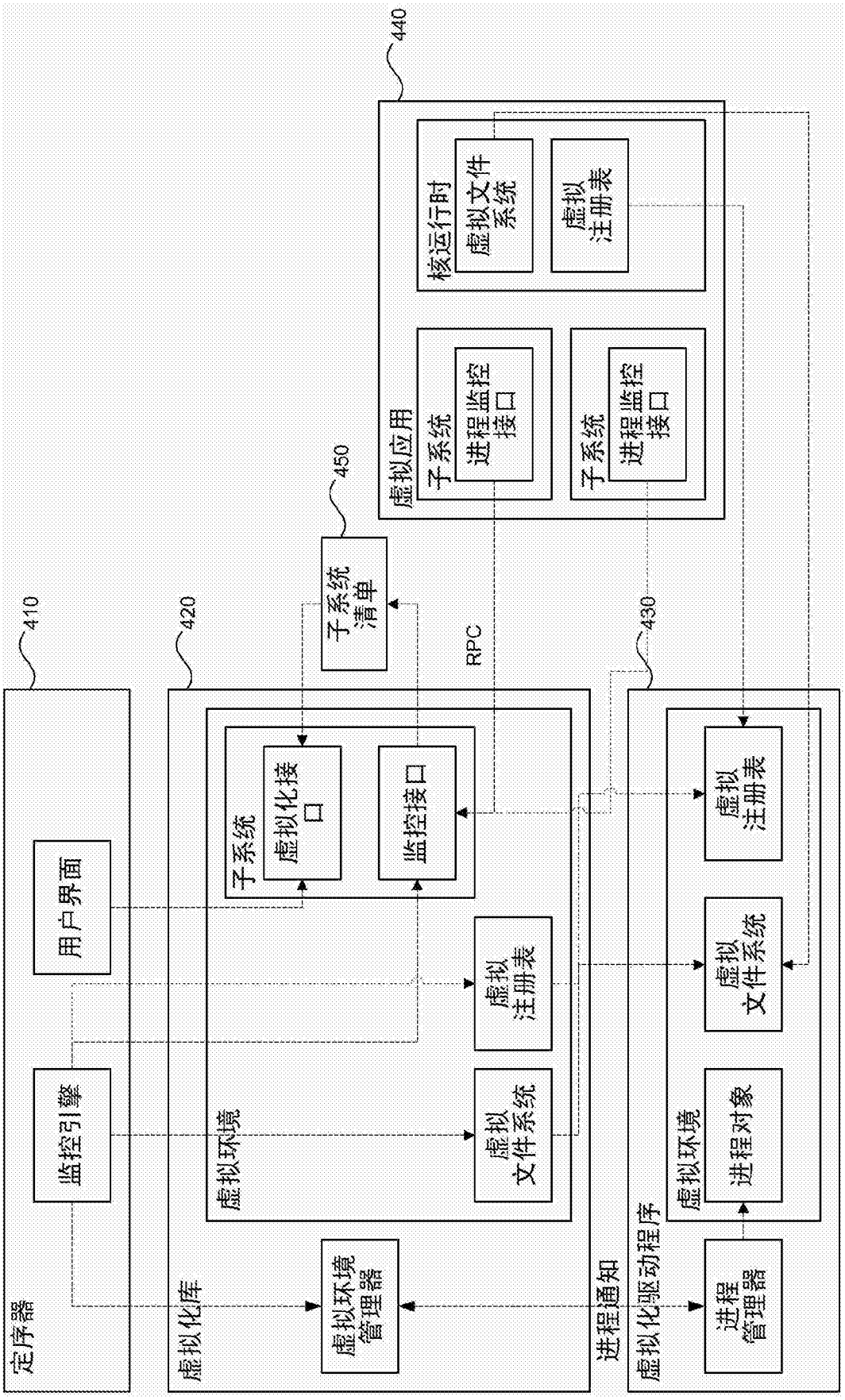

440
核运行时
虚拟文件系统
虚拟注册表
虚拟应用
子系统
进程监控接口
子系统
进程监控接口
450
子系统清单
410
420
430
RPC
用户界面
监控引擎
定序器
子系统
虚拟化接口
监控接口
虚拟注册表
虚拟文件系统
虚拟环境
虚拟环境管理器
虚拟化库
进程通知
虚拟注册表
虚拟文件系统
虚拟环境
进程对象
进程管理器
虚拟化驱动程序

图 4